**Source: Tencent**

**Title: AI/ML model format: PyTorch**

**Document for: Agreement**

**Agenda Item: 9.7**

# Introduction

In this contribution, we introduce PyTorch as an example format for AI/ML model distribution. PyTorch is an open-source machine learning framework developed by Facebook's AI research team. It provides a range of tools for building and training machine learning models, as well as support for deploying these models in various formats.

1. **Overview of PyTorch**

PyTorch is based on the concept of tensors, which are multi-dimensional arrays of numerical data. Similarly to TensorFlow, Tensors are a fundamental data structure used in PyTorch to represent the input data and the parameters of the machine learning model. PyTorch provides a range of operations for manipulating tensors, such as addition, multiplication, and convolution.

PyTorch also supports dynamic computation graphs, which allow for more flexibility in building and training machine learning models. This means that the computational graph can be modified on-the-fly during runtime, which makes it easier to build complex models and experiment with different architectures. Additionally, PyTorch provides a high-level API called TorchScript, which allows for models to be exported to a portable format that can be executed on various platforms.

1. **PyTorch for model distribution**

PyTorch provides several formats for distributing machine learning models, such as PyTorch JIT (Just-In-Time) and TorchScript. PyTorch JIT allows for models to be compiled on-the-fly, which provides performance benefits for large models or when deploying to resource-constrained environments. TorchScript allows for models to be exported to a portable format that can be executed on various platforms, such as mobile devices, web browsers, and embedded systems.

PyTorch also supports ONNX (Open Neural Network Exchange), which is an open format for exchanging machine learning models between different frameworks. ONNX allows for models to be trained in PyTorch and then exported to be executed in other frameworks, such as TensorFlow or Caffe2

NOTE: it is expected to analyze:

* The different distribution AI/ML formats that can be used with the PyTorch framework.
* The impacts of the selection of PyTorch framework in terms of interoperability of the corresponding AI/ML formats.
1. **Main differences with Tensorflow**

**Computational graph:** TensorFlow uses a static computational graph, which means that the graph is defined and compiled before the training begins. On the other hand, PyTorch uses a dynamic computational graph, which allows for more flexibility in building and modifying the graph during runtime.

**Ease of use:** PyTorch is generally considered to be more user-friendly and simpler than TensorFlow. This is partly due to its dynamic computational graph, which makes it easier to experiment with different models and architectures. PyTorch also has a more Python-like syntax, which is familiar to many developers.

**Visualization:** TensorFlow provides a comprehensive visualization tools, which allows users to monitor the training progress and visualize the model's performance. PyTorch does not have a built-in visualization tool, but there are several third-party libraries available, such as PyTorch Lightning and Visdom.

**Ecosystem:** TensorFlow has a larger ecosystem than PyTorch, with more resources and community support. TensorFlow also has better support for deploying models on mobile devices and in production environments. However, PyTorch has been gaining popularity in recent years and has a growing ecosystem.

**Research:** PyTorch is more popular in the research community, as it allows for faster prototyping and experimentation due to its dynamic computational graph. TensorFlow is more commonly used in industry for production-level applications due to its static graph and better support for deployment.

1. **Conclusion**

It is proposed to list PyTorch as part of the available frameworks for AI/ML model distribution in the Permanent document.